

Secure Coding in C and C++ (SECC-CCPP)

ID SECC-CCPP Price 2,250.— €(excl. tax) Duration 3 days

Course Overview

Your application written in C and C++ works as intended, so you are done, right? But did you consider feeding in incorrect values? 16Gbs of data? A null? An apostrophe? Negative numbers, or specifically -1 or -231? Because that's what the bad guys will do – and the list is far from complete.

Handling security needs a healthy level of paranoia, and this is what this course provides: a strong emotional engagement by lots of hands on labs and stories from real life, all to substantially improve code hygiene. Mistakes, consequences, and best practices are our blood, sweat and tears.

All this is put in the context of C and C++, and extended by core programming issues, discussing security pitfalls of these languages.

So that you are prepared for the forces of the dark side.

So that nothing unexpected happens.

Nothing.

Who should attend

C/C++ developers

Prerequisites

General C/C++ development

Course Objectives

- Getting familiar with essential cyber security concepts
- Identify vulnerabilities and their consequences
- Learn the security best practices in C and C++
- Input validation approaches and principles

Course Content

- Cyber security basics
- Buffer overflow
- Memory management hardening
- Common software security weaknesses
- Wrap up

Detailed Course Outline

DAY 1

Cyber security basics

- What is security?
- Threat and risk
- Cyber security threat types
- Consequences of insecure software
 - Constraints and the market
 - The dark side

Buffer overflow

- Assembly basics and calling conventions
 - x64 assembly essentials
 - Registers and addressing
 - Most common instructions
 - Calling conventions on x64
 - Calling convention – what it is all about
 - The stack frame
 - Stacked function calls
- Memory management vulnerabilities
 - Memory management and security
 - Vulnerabilities in the real world
 - Buffer security issues
 - Buffer overflow on the stack
 - Buffer overflow on the stack – stack smashing
 - Exploitation – Hijacking the control flow
 - Lab – Buffer overflow 101, code reuse
 - Exploitation – Arbitrary code execution
 - Injecting shellcode
 - Lab – Code injection, exploitation with shellcode
 - Buffer overflow on the heap

- Unsafe unlinking
- Case study – Heartbleed
- Pointer manipulation
 - Modification of jump tables
 - Overwriting function pointers
- Best practices and some typical mistakes
 - Unsafe functions
 - Dealing with unsafe functions
 - Lab – Fixing buffer overflow
 - What's the problem with `asctime()`?
 - Lab – The problem with `asctime()`
 - Using `std::string` in C++
 - Unterminated strings
 - `readlink()` and string termination
 - Manipulating C-style strings in C++
 - Malicious string termination
 - Lab – String termination confusion
 - String length calculation mistakes
 - Off-by-one errors
 - Allocating nothing

DAY 2

Memory management hardening

- Securing the toolchain
 - Securing the toolchain in C and C++
 - Compiler warnings and security
 - Using `FORTIFY_SOURCE`
 - Lab – Effects of `FORTIFY`
 - AddressSanitizer (ASan)
 - Using AddressSanitizer (ASan)
 - ASan changes to the prologue
 - ASan changes to memory read/write operations
 - ASan changes to the epilogue
 - Lab – Using AddressSanitizer
 - Stack smashing protection
 - Detecting BoF with a stack canary
 - Argument cloning
 - Stack smashing protection on various platforms
 - SSP changes to the prologue and epilogue
 - Lab – Effects of stack smashing protection
 - Address Space Layout Randomization (ASLR)
 - ASLR on various platforms
 - Lab – Effects of ASLR
 - Circumventing ASLR – NOP sleds
 - Non-executable memory areas
 - The NX bit
 - Write XOR Execute (W^X)
 - NX on various platforms
 - Lab – Effects of NX

- NX circumvention – Code reuse attacks
 - Return-to-libc / arc injection
- Return Oriented Programming (ROP)
 - Protection against ROP

Common software security weaknesses

- Security features
 - Authentication
 - Authentication basics
 - Multi-factor authentication
 - Authentication weaknesses – spoofing
 - Case study – PayPal 2FA bypass
 - Password management
 - Inbound password management
 - Storing account passwords
 - Password in transit
 - Lab – Is just hashing passwords enough?
 - Dictionary attacks and brute forcing
 - Salting
 - Adaptive hash functions for password storage
 - Password policy
 - NIST authenticator requirements for memorized secrets
 - Case study – The Ashley Madison data breach
 - The dictionary attack
 - The ultimate crack
 - Exploitation and the lessons learned
 - Password database migration
 - Outbound password management
 - Hard coded passwords
 - Best practices
 - Lab – Hardcoded password
 - Protecting sensitive information in memory
 - Challenges in protecting memory
 - Heap inspection
 - Compiler optimization challenges
 - Lab – Zeroization challenges
 - Sensitive info in non-locked memory
- Code quality
 - Data handling
 - Type mismatch
 - Lab – Type mismatch
 - Initialization and cleanup
 - Constructors and destructors

- Initialization of static objects
- Lab – Initialization cycles
- Array disposal in C++
- Lab – Mixing delete and delete[]
- Memory and pointers
 - Memory and pointer issues
 - Pointer handling pitfalls
 - Pointer usage in C and C++
 - Use after free
 - Lab – Use after free
 - Lab – Runtime instrumentation
 - Double free
 - Memory leak
 - Smart pointers and RAI
 - Smart pointer challenges

- Case study – The Stockholm Stock Exchange
- Lab – Signed / unsigned confusion
- Integer truncation
- Lab – Integer truncation
- Case study – WannaCry
- Best practices
 - Upcasting
 - Precondition testing
 - Postcondition testing
 - Using big integer libraries
 - Best practices in C
 - UBSan changes to arithmetics
 - Lab – Handling integer overflow on the toolchain level in C/C++
 - Best practices in C++
 - Lab – Integer handling best practices in C++

DAY 3

Common software security weaknesses

- Input validation
 - Input validation principles
 - Blacklists and whitelists
 - Data validation techniques
 - What to validate – the attack surface
 - Where to validate – defense in depth
 - How to validate – validation vs transformations
 - Validation with regex
 - Injection
 - Injection principles
 - Injection attacks
 - Code injection
 - OS command injection
 - Lab – Command injection
 - OS command injection best practices
 - Avoiding command injection with the right APIs
 - Lab – Command injection best practices
 - Case study – Shellshock
 - Lab – Shellshock
 - Process control – library injection
 - DLL hijacking
 - Lab – DLL hijacking
 - Integer handling problems
 - Representing signed numbers
 - Integer visualization
 - Integer promotion
 - Integer overflow
 - Lab – Integer overflow
 - Signed / unsigned confusion

- Files and streams
 - Path traversal
 - Path traversal-related examples
 - Lab – Path traversal
 - Path traversal best practices
 - Lab – Path canonicalization
- Format string issues
 - The problem with printf()
 - Lab – Exploiting format string
- Time and state
 - Race conditions
 - File race condition
 - Time of check to time of usage – TOCTTOU
 - Lab – TOCTTOU
 - Insecure temporary file

Wrap up

- Secure coding principles
 - Principles of robust programming by Matt Bishop
 - Secure design principles of Saltzer and Schröder
- And now what?
 - Software security sources and further reading
 - C and C++ resources

About Fast Lane



Fast Lane is a global, award-winning specialist in technology and business training as well as consulting services for digital transformation. As the only global partner of the three cloud hyperscalers- Microsoft, AWS and Google- and partner of 30 other leading IT vendors, Fast Lane offers qualification solutions and professional services that can be scaled as needed. More than 4,000 experienced Fast Lane professionals train and advise customers in organizations of all sizes in 90 countries worldwide in the areas of cloud, artificial intelligence, cyber security, software development, wireless and mobility, modern workplace, as well as management and leadership skills, IT and project management.

Fast Lane Services

- ✓ High End Technology Training
- ✓ Business & Soft Skill Training
- ✓ Consulting Services
- ✓ Managed Training Services
- ✓ Digital Learning Solutions
- ✓ Content Development
- ✓ Remote Labs
- ✓ Talent Programs
- ✓ Event Management Services

Training Methods

- ✓ Classroom Training
- ✓ Instructor-Led Online Training
- ✓ FLEX Classroom – Classroom & Online Hybrid
- ✓ Onsite & Customized Training
- ✓ E-Learning
- ✓ Blended & Hybrid Learning
- ✓ Mobile Learning

Technologies & Solutions

- ✓ Digital Transformation
- ✓ Artificial Intelligence
- ✓ Cloud
- ✓ Networking
- ✓ Cyber Security
- ✓ Wireless & Mobility
- ✓ Modern Workplace
- ✓ Data Center



Worldwide Presence
with high-end training centers
around the globe



Multiple Awards
from vendors such as AWS,
Microsoft, Cisco, Google, NetApp,
VMware



Experienced SMEs
with over 19.000 combined
certifications

Germany

**Fast Lane Institute for Knowledge
Transfer GmbH**
Tel. +49 40 25334610
info@flane.de / www.flane.de

Austria

ITLS GmbH
(Partner of Fast Lane)
Tel. +43 1 6000 8800
info@itls.at / www.itls.at

Switzerland

**Fast Lane Institute for Knowledge
Transfer (Switzerland) AG**
Tel. +41 44 8325080
info@flane.ch / www.flane.ch